BEST AVAILABLE IMAGES

Defective images within this document are accurate representation of The original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

Refine Search

Search Results -

Terms	Documents
L6 and (respon$ or return$)	1

Database:
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Search: L8

Refine Search | Recall Text | Clear | Interrupt

Search History

DATE: Wednesday, February 11, 2004 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; THES=ASSIGNEE; PLUR=YES; OP=OR			
L8	L6 and (respon$ or return$)	1	L8
L7	L6 and (bill$ or charg$ or cost$)	1	L7
L6	5638448.pn.	1	L6
L5	L3 and encrypt$	1	L5
L4	L3 and (first same protect$)	1	L4
L3	5634012.pn.	1	L3
L2	L1 or 5634012.pn.	3	L2
L1	5485330.pn. or 5308336.pn.	2	L1

102 157 3689 10 4 −3

END OF SEARCH HISTORY

h e b b cg b e e ch

First Hit Fwd Refs
End of Result Set

Generate Collection Print

L8: Entry 1 of 1 File: USPT Jun 10, 1997

DOCUMENT-IDENTIFIER: US 5638448 A
TITLE: Network with secure communications sessions

Detailed Description Text (7):
To use resources on the network, users must first logon the server to prove their identity. A logon request is sent from the client's logon application to the requester on the client computer. Before logon data can be exchanged between the applications and the requester, a command manager is created by the requester to accept application requests. The command manager is responsible for housekeeping requests within the client computer.

Detailed Description Text (22):
5- The random numbers Ra and the CRC signatures are then decrypted. The server calculates the CRC signature of the packet header, the user ID and the random number Ra. If the calculated signatures match the decrypted signatures C1 and C2 stored in the packet, and if password Ka matches Kb, the server manipulates the client random number Ra with a predefined formula, generates a random number Rb, and encrypts both random numbers Ra and Rb with the password Kb before sending the first logon response packet to the client.

Detailed Description Text (24):
1- The client decrypts the first logon response packet.

Detailed Description Text (25):
2- The client manipulates the random number Ra with the predefined formula and compares it with the one returned from the server. If the numbers match, the client knows that it is connected to the correct server, not a fraud server from which an eavesdropper has captured transmissions from the previous logon and is echoing packets back to the client computer.

Detailed Description Text (35):
1- The second logon response packet is decrypted by the client.

Detailed Description Text (66):
The heart of this authentication procedure is in the middle part of the logon packet, which contains the random number Ra and the CRC signatures. Since the CRC signature of the random number Ra is encrypted and sent along with the logon packet, the server can authenticate the user right on the first logon packet. The manipulation of the random numbers Ra and Rb in the challenge-response fashion is to help the server defeat the replaying of the logon packet and to allow the client to authenticate the server and to defeat packet replaying as well.

Detailed Description Text (68):
In addition, the server name is isolated from the user ID and password when creating a one-way hashed password to allow the portability of the database. For example, when a business grows, another server may be needed at another location and the database can be easily transferred to the new server. Of course, it would be less time-consuming to delete unauthorized users from the database than to add

h e b b g e e e f c e f eb e ge

authorized users to the new one. To better protect the valuable information in the database, a password is required before access to the database is granted. More important, the database can be shared among servers. For example, a server Sb can receive the first logon packet and forward the user ID to a database server Sc within a private network for verification. If an access record is found and the user can access the server Sb at this date and time, the database server Sc returns the encrypted one-way hashed password Kb to the server Sb. The server Sb then continues the challenge-response as if the password Kb is returned from a local database. Note that the database server Sc can encrypt the one-way hashed password Kb with the session key defined for communication between the server Sb and Sc before sending it across the private network is security is desired.

Detailed Description Text (70):
Finally, the logon protocol of the preferred embodiment is more suitable for a client/server distributed environment, because this logon protocol allows both client and server to authenticate each other without sending the user password across the communication media and prevent intruders from deleting, inserting, modifying, or replaying the logon packets. In addition, if the logon procedure fails at any point, the server releases all resources and destroys the connection without sending the response packet at that point, i.e., if the user enters a wrong server name in the very first logon packet, nothing is sent out from the server to prevent the user, a potential intruder, from knowing anything about the server. Note that this mutual authentication technique requires the client machine to have a local CPU so that the password will not be transmitted over the network before being encrypted.

Detailed Description Text (74):
If the resource is local, the request router 106 calls a local system function call to perform the request and returns the control to the application 102. However, if the resource is remote, the request router 106 first searches its local list to see if the needed communication handle is already stored in the list. This communication handle contains information of the read 204 and write 208 tokens (shown in FIG. 2) and their associated resources. If the communication handle is not found in the local list, the request router 106 sends a message to the requester 110 over the request channel 112 to obtain the handle. Once the handle is obtained, the request router 106 creates a response signal, i.e., a return address, requests the ownership of the write token 208, stores the response signal into the packet header, builds a packet based on the application's 102 request into the write token 208, and signals the session write thread 206 of the communication channel 114 that there is a packet to send.

Detailed Description Text (75):
If the application data is larger than the packet capacity, the request router 106 can send multiple packets in a series at this point. After the packet is sent to the server, the request router releases the write token for use by another thread in the same process or a different process. If the packet was sent to the server successfully, the request router 106 waits for the corresponding response packets, i.e., a packet can cause multiple response packets returned from the server.

Detailed Description Text (76):
When a response packet arrives, the session read thread uses the response signal to tell the corresponding request router that its response packet has come and is available in the read token. At that time, the read token is accessed exclusively by the designated request router. The router then transfers data in the response packet directly to the application's buffers and signals the session read thread 202 of the communication channel 114 that the read token 206 is no longer in use so that the session read thread 202 can re-use the read token 206 for other incoming packets. Finally, after all response packets of a request packet have arrived, the request router 106 destroys the response signal and returns control to the application 102. The final response packet is determined by a bit in the packet

attribute.

Detailed Description Text (79):
The request router 106 can also perform any preparation necessary to transfer the application 102 request to the requester 110 before requesting the ownership of the write token 208 to reduce the time it takes to access the write token 208. In addition, the request router 106 remembers resources for one application 102 at a time. Thus, it reduces the time to search for the needed information. With this method of sending and receiving packets, data can be exchanged asynchronously between a client and a server with minimum resources in a minimum time. In addition, request packets can be accumulated on the server for processing while the previous response packet is processed by the communication devices 120, 122, 124, 126 or traveling over the network.

Detailed Description Text (83):
When the Requester 110 performs automatic logon due to a request from a Request Router 106 for a communication handle, the Command Manager 140 searches the Global Mounting Table 132 to find an appropriate server name. If the server name cannot be found, the user has not performed logon manually. However, if the server name is found, the Command Manager 140 will create a Session Write Thread 206 and its associated resources, decrypt the encrypted key Ka and store logon information such as the user ID, the key Ka, and the server name into the Session Info Block as shown in FIG. 5A-B. The auto-logon procedure is then performed by the Session Write Thread 206. After the logon procedure is successfully completed, the Session Write Thread 206 will automatically create the Session Read Thread 204 as described earlier. A return code is then returned to the Command Manager 140 of the Requester to indicate success or failure. If the auto-logon is successful, the Requester 110 returns the communication handle to the Request Router 106. Otherwise, an error code is returned to the Request Router 106. Compression engines (CE) 138 are used to compress data for performance reasons.

Detailed Description Text (85):
FIG. 3A and B is a flowchart which illustrates the transfer of information in a session after the logon procedure has completed. When a resource request 302 is made, the system 304 first tests to see if it is for a local resource 304. If so, a local function is called 312 and control is returned 310 to the application. If it is not a local resource, the system creates a response signal 306. If the response signal 306 cannot be created, control is returned to the application. If it is, then the local list is searched 314 for the communication handle. If the communication handle is not found 316, a communication handle is obtained 318 from the requester and then ownership if the write token is requested 320. However, if the communication handle is found 316, then ownership if the write token is immediately requested 320.

Detailed Description Text (86):
If no error occurs when the request for ownership of the write token is made 322, then the response signal is stored in the packet header 326, a request packet is built into the write token 328, the write thread sends the packet, and the write token is released 332. If an error is detected when the packet is sent, the response signal is destroyed 342 and control is returned 344 to the application. If no errors occur during packet transmission 344, then the system waits 336 for the response packet, the data in the response packet is transferred 338 into the application's buffer, the read token is released 340, the response signal is destroyed 342 and control is returned 344 to the application.

Detailed Description Text (112):
FIG. 12 is a more detailed view of the server 1004 of FIG. 11. A control manager 1122 within the server 1004 is responsible for communication between the server 1004 and other applications on the server 1004 machine. Thus, the server 1004 can be informed if a database has been changed by a resource control application. The

h e b b g e e e f c e f eb e ge

server 1004 can also accept a message from another application 102 to send to all or selected clients over active sessions. If an electronic mail system should be needed, the server 1004 can save the message and wait until a client is logged on to send the message over the session. To support these features, the control manager 1122 posts message or e-mail packets to the incoming packet queues 1206 of the sessions 1120. When the server processing threads 1114, 1116 of the sessions 1120 retrieves the packets from the queue 1206, it will process the packets based on the packet types defined in the packet headers.

Detailed Description Text (115):
The requester also has the capability to signal request routers 106 of all applications 102 when a communication session is terminated abnormally whether the request routers 106 are sending request packets or waiting on response packets. In order to perform this feature, the response signals (i.e., the return addresses stored in the request packets) are saved in response-signal queues by the session write thread 1116. Each communication session has a response-signal queue 1206 to reduce the search time. When the response packets are successfully delivered, their corresponding response signals are removed from the queue by the session read threads 1114 of the corresponding communication channels. If an application 102 terminates before its response packets arrive, the response packets are discarded and the response signals are also removed from the queue after all chaining response packets have arrived.

Detailed Description Text (116):
In addition, the read thread of the client session also recognizes different types of packets to determine whether it should route the received packets to the application's request router or to a message manager within the requester. The message manager of the requester is responsible for message and e-mail packets sent from the connected servers. This feature is important because it allows the server to initiate the sending of packets while a session is active. As an example, a hot-link can be defined so that a server can inform the connected clients if a database should be changed or a server administrator can send a message to all or selected clients telling them if a server should be out of service shortly, etc. In a more advanced application, an electronic-mail server application can be written so that the message packets are saved on the server until a client is logged on. At that time, the server will send the saved messages to the connected client.

Detailed Description Text (117):
In the prior art, the requester is the one that translates and formats requests from the applications; thus, it cannot perform preparation ahead of time. In addition, information accumulating in one place could increase the search time. The prior art requires an intrinsics modules in both the application and the requester which may require more resources to be allocated and more machine instructions to be executed. Furthermore, the prior art does not have the capability to accumulate multiple request packets from a requester so that the server can process the next packet request while the previous response packet is traveling back to the requester on the network or being processed by communication devices in their own memory buffers.

Detailed Description Text (118):
In contrast to the prior art, the preferred embodiment contains the formatting and translating code in just one place, the request router 106. The requester only compresses and/or encrypts packet headers and packet data if necessary and then calls the transport functions to send the packets to the server. In addition, requester 110 is also responsible for saving logon and mounting information, managing the communication sessions, and delivering response packets received from multiple network domains to multiple request routers while sending request packets to the multiple network domains. Requester 110 does not need to know the format of the response data, and can deliver the response packets immediately upon receiving them. The request routers 106 can then format or translate the response data in the

applications time slices while the requester 110 is waiting for other incoming response packets or reading data from the communication devices 120, 122, 124, 126. Thus, the preferred embodiment achieves better performance than the prior art.

Detailed Description Text (119):
The prior art also requires the intrinsic modules to translate and format the application data from a program stack segment to a parameter block before sending it to its requester where the data is once again formatted or copied into a data communication buffer. In contrast, the request routers 106 in the preferred embodiment format the application data only once and store the formatted data into the write token which will be used by the requester and the communication subsystem to send the request packets to the server. When the response packets arrive, the requester 110 uses the response signals to tell the corresponding request routers that their response packets have arrived. At that time, the request routers 106 transfer response data directly from the read tokens into the application buffers. Thus, the preferred embodiment eliminates the overhead of copying data between memory buffers.

Detailed Description Text (124):
Note that in the very first logon manually performed by the user, the operation is slightly different than the auto-logon mentioned in the above paragraph. The requester first receives a logon request from the logon application, it establishes the session itself and then performs the logon. This is so done by the command manager of the requester, not by the session manager. The session manager is responsible for dropping the session if no data is transmitted for a certain period of time.

Detailed Description Text (125):
Since request packets are accumulated in the packet queue in the preferred embodiment, the request packets may not be processed immediately upon arrival. In contrast, the prior art must process the request packets immediately to return the status or data to the requester. This may indicate that other applications on the client computer must wait until the return packet has arrived and processed before they can send their requests to the same host computer.

Detailed Description Text (127):
As shown earlier, the very first logon packet is encrypted with three different keys for different parts of the packet. The header of the logon packet is encrypted with a key generated from the server name. This is design to detect outside intruders early in the verification process. For intruders working inside an organization, the server name may be known. Then it comes the middle part of the logon packet which contains the 64-bit random number and the CRC value. These are the heart of the verification since it is encrypted with the key generated from the user ID and the secret password. This scheme allows the server to detect the intruding logon right on the very first packet. The challenge-response process that following the logon packet is to defeat re-played packets.

Detailed Description Text (138):
After the client decrypts the first logon response packet, it manipulates the random number Ra with the predefined formula and compares it against the one returned from the server. If the numbers match, the client knows that it communicates with a correct server, not a fraud server where an eavesdropper has set up to echo back captured packets. The client now manipulates the random number Rb with another predefined formula to form Rb' and concatenates it with the client's initiating data, i.e., the client initial packet sequence number and the encryption mode for the session, to form a second logon packet. The client then masks the initial data with the random number Ra to hide the possibly known text, encrypts the third logon packet and sends it to the server. Note that the client starts to send its initial data to the server only if the random number Ra' is verified to ensure that the server is a trusted party.

Detailed Description Text (173):
Other performance enhancements can be realized by sending multiple requests from a client system to a server in a single transmission. In addition, system performance can also be improved by preventing subsequent packet data from being transmitted between a server and a client until the previous packet data has been responded to Requestors can batch application work within the client machine and communicate with the remote server resulting in remote batching. As a result, the requestor acts as a remote server function. This function is done asynchronously to enhance performance.

CLAIMS:

15. A method, as in claim 14, wherein succeeding data packets from a client are continuously sent to the server via a write thread and responses from the server are continuously received via a read thread.

17. A method, as in claim 9, wherein succeeding data packets from a client are continuously sent to the server via a write thread and responses from the server are continuously received via a read thread.